VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Asistenční aplikace pro zákazníky autobazarů

Assistance Application for Pre-Owned Car Customers

2014 **Vít Juřica**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Vít Juřica**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Asistenční aplikace pro zákazníky autobazarů
Assistance Application for Pre-Owned Car Customers

Zásady pro vypracování:

Cílem bakalářské práce je návrh a implementace mobilní aplikace umožňující zákazníkům snadnější orientaci na trhu s ojetými vozy. Základem aplikace bude serverová komponenta automaticky procházející všechny nabídky zveřejňované na inzertních serverech a ukládající je do databáze (podle možností parsovat HTML/XML data z inzertních serverů). Během provozu tak vznikne databáze jak nabízených, tak i již prodaných vozů. Uživatel bude do této databáze přistupovat pomocí mobilní aplikace pro OS Android. Umožní mu vyhledávání v aktuálních inzerátech, zobrazí historii cen stejného, nebo podobného vozidla, vývoj cen a případně zobrazí informace o předchozím prodeji.

1. Návrh a implementace databáze pro ukládání dat.
2. Realizace automatického robota (crawler) pro její plnění.
3. Návrh a implementace klientské aplikace pro OS Android.
4. Testy databázové a aplikační části.
5. Shrnutí dosažených výsledků.

Seznam doporučené odborné literatury:

[1] James Steele, Nelson To, The Android Developer's Cookbook: Building Applications with the Android SDK, Addison-Wesley Professional, 2010, ISBN-13: 978-0321741233
[2] Reto Meier, Professional Android 4 Application Development, Wrox, 2012, ISBN-13: 978-1118102275
[3] Sayed Hashimi, Pro Android 2, Apress, 2010, ISBN-13: 978-1430226598

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Ing. Michal Krumnikl**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 2. 5. 2014

Poděkování

Rád bych poděkoval Ing. Michalu Krumniklovi za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Tématem této bakalářské práce je mobilní aplikace, pro operační systém Android, která umožňuje snadnější orientaci na trhu s ojetými vozidly. V první části bakalářské práce je popis podobných aplikací a implementace databáze. V další části je popsána implementace automatického robota pro plnění databáze v jazyce Python. Následující část je věnována samotné mobilní aplikaci určenou k vyhledávání vozidel. Aplikace má název Pre-Owned Cars.

Klíčová slova

Android, PHP, Python, Scrapy, JSON, crawler, ojetá auta

Abstract

The topic of this bachelor thesis is a mobile application for the Android operating system, which makes it easier to focus on the used car market. In the first part of the thesis there is a description of similar applications and database implementation. The next section describes the implementation of automatic robot for filling database in Python. The following section is dedicated to the mobile application designed to search vehicles. The application is called the Pre-Owned Cars.

Key words

Android, PHP, Python, Scrapy, JSON, crawler, used cars

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
GPL	General Public License	Všeobecná veřejná licence
JSON	JavaScrip Object Notation	JavaScriptový objektový zápis
AVD	Android Virtual Device	Virtuální zařízení s Android
XML	Extensible Markup Language	Rozšířitelný značkovací jazyk
ADT	Android Developer Tools	Vývojové nástroje pro Android
SDK	Software Development Kit	Sada pro vývoj softwaru
URL	Uniform Resource Locator	Jednotný vyhledávač zdrojů
AJAX	Asynchronous JavaScript and XML	Asynchronní JavaScript a XML
API	Application Programming Interface	Rozhraní pro programování aplikací

Obsah

1 Úvod

Čím dál tím více lidí vlastní chytrý telefon a díky možnostem operačního systému Android mohou využít svůj telefon téměř ve všech oblastech života. Systém android je Android nabízí pro vývojáře nespočet možností, které usnadňují vývoj aplikace a díky obchodu Google Play je snadné aplikace prezentovat.

Cílem této bakalářské práce je vytvořit aplikaci pro vyhledávání ojetých aut z inzertních serverů, zobrazení historie a vývoj ceny aut. Pro tuto bakalářskou práci jsem zvolil server Sauto.cz, který obsahuje velké Toto zahrnovalo vytvoření fungujícího systému, který obsahuje komponentu pro procházení inzertních serveru s ojetými vozidly a jejich ukládání do databáze. Do této databáze bude uživatel přistupovat pomocí mobilní aplikace.

Na začátku práce jsem popsal aplikace, zaměřující se na ojetá vozidla, abych zjistil, jaké prvky aplikace potřebuje, a které jsou naopak zbytečné. Dále jsem se zaměřil na popis struktury databáze a PHP skriptů pro získávání dat z databáze.

V další části práce jsem se zaměřil na automatického robota, napsaného v jazyce Python, tzv. crawler. Který slouží k získávání informaci z inzertních serverů a jejich ukládání do databáze. A popis mobilní aplikace, kde jsou popsány všechny její funkce.

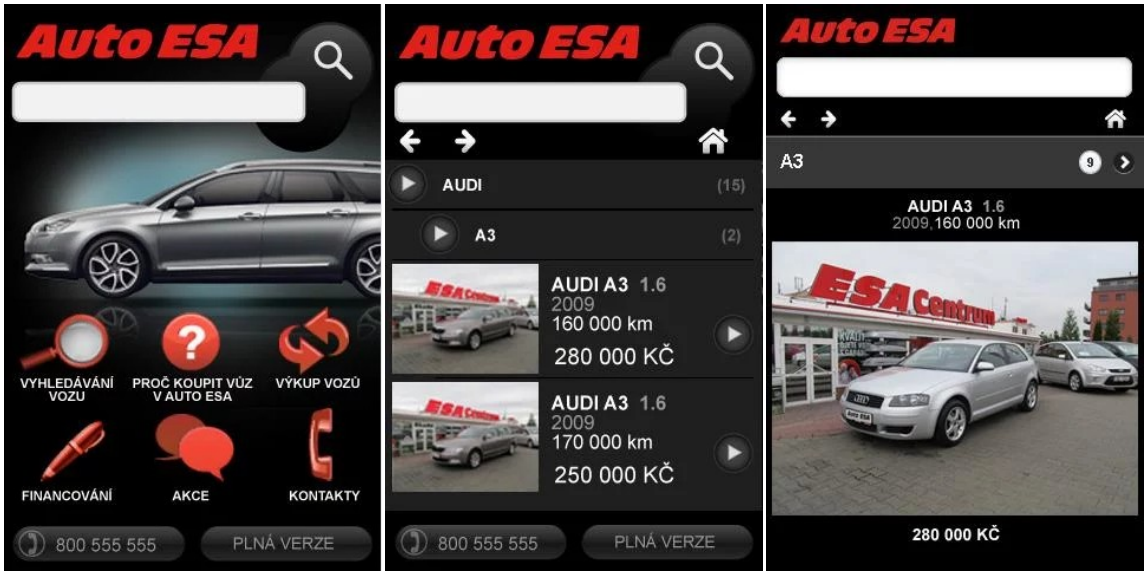
V poslední části jsem provedl testy databázové a aplikační části.

2 Srovnání podobných aplikací

V následujících oddílech jsou popsány mobilní aplikace, které se se svým zaměřením a funkcemi podobají aplikaci vytvářené v této bakalářské práci. Informace byly čerpány z vlastní zkušenosti a z komentářů na [1].

2.1 AUTO ESA

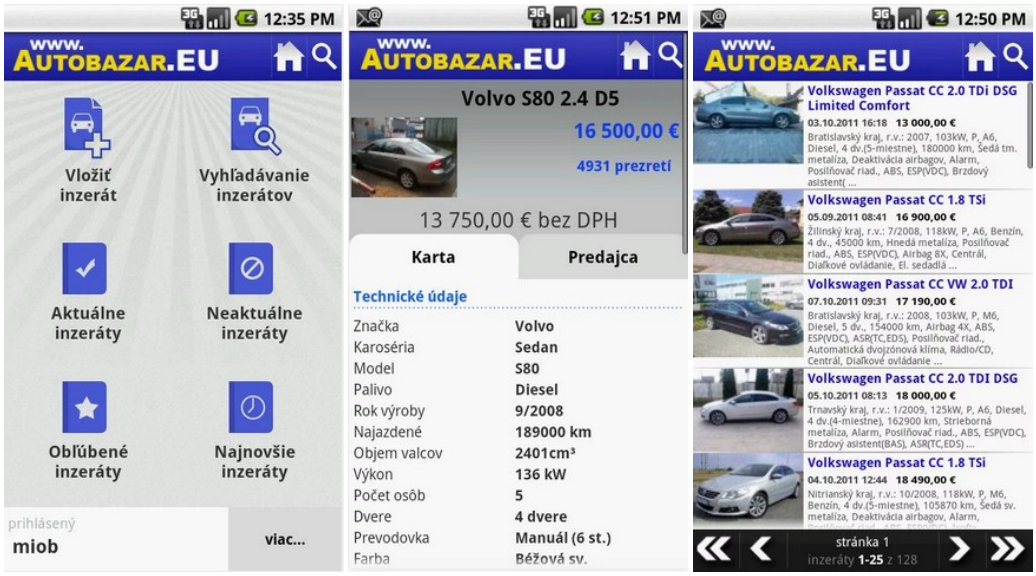
Aplikace AUTO ESA je domovskou aplikací autobazaru Auto ESA. Vyhledávání v aplikaci je možné pouze podle názvu vozidla, tudíž nelze vyhledat vozidlo podle roku výroby nebo stavu tachometru. Navigace skrze aplikaci není úplně v pořádku, často se stává, že při přechodu na předchozí okno aplikace zobrazí úplně jinou obrazovku. Protože se jedná o domovskou aplikaci autobazaru, lze na kartě vozidla provést Nezávaznou rezervaci automobilu. Aplikace dále obsahuje kontaktní formulář pro výkup vozu a obsahuje informace o finančních službách.



Obrázek 1.1: Uživatelské rozhraní aplikace Auto ESA

2.2 Autobazar EU

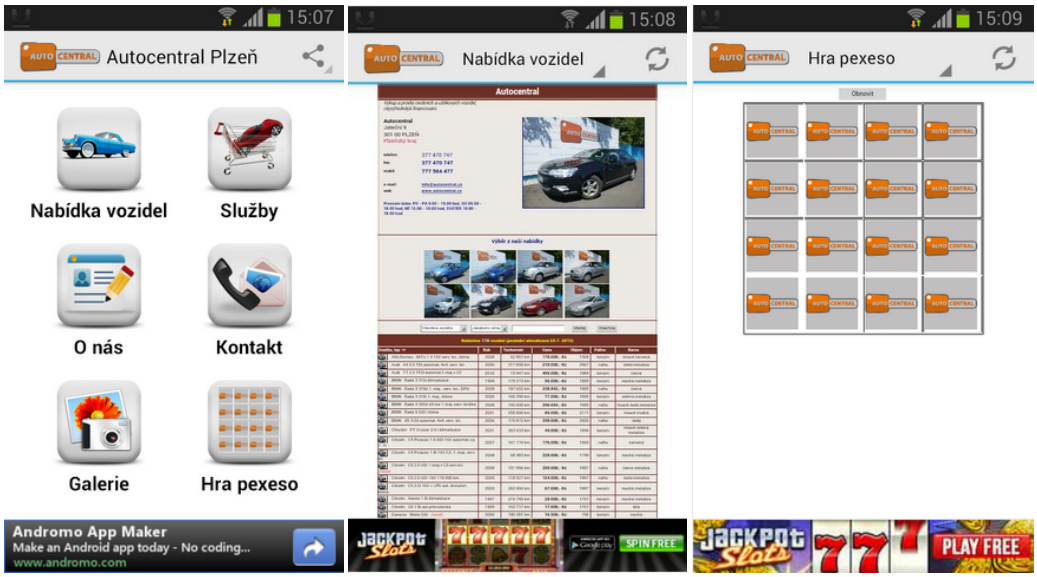
V aplikaci Autobazar EU lze vkládat inzerát, zobrazovat aktuální, neaktuální, nejnovější a oblíbené inzeráty, ale je potřeba být přihlášen. Bohužel z aplikace není možnost registrace. Vyhledávání inzerátů lze podle fulltextového vyhledávání s našeptávačem, nebo podle parametrů. Na kartě vozidla lze kontaktovat prodejce, najít prodejce na mapě a odeslat odkaz na vozidlo známému.



Obrázek 1.2: Uživatelské prostředí aplikace Autobazar EU

2.3 Autocentral Plzeň

Tato aplikace je domovskou aplikací firmy Autocentral Plzeň. Neobsahuje žádné vyhledávání vozidel. Pouze zobrazí seznam všech automobilů v tabulce, což velmi znepříjemňuje procházení vozidel. Aplikace obsahuje seznam služeb, které otevřou nové okno prohlížeče a zobrazí internetové stránky firmy. Dále jsou zde uvedeny informace o firmě, kontakt a obsahuje také hru pexeso.



Obrázek 1.3: Uživatelské prostředí aplikace Autocentral Plzeň

3 Návrh a implementace databáze pro ukládání dat

Vyvíjený systém se skládá ze tří částí. V první části je implementován automatický robot, který prochází internetové stránky inzertního serveru www.sauto.cz a extrahuje z nich informace. Princip automatického robota je popsán v kapitole číslo čtyři. Druhá část je databáze, která slouží pro ukládání informací a je popsána v této kapitole. Poslední částí je mobilní aplikace, která zobrazuje informace uložené v databázi. Implementace mobilní aplikace je popsána v kapitole pět.

Implementovaná databáze slouží k ukládání dat o vozidlech vystavených na inzertních serverech. S informacemi uloženými v databázi je možné zjistit vlastnosti vozidla a jeho stav, lze také dohledat historii vozidla i jeho ceny a další informace. Databáze je implementována na lokálním i na internetovém serveru, typ databáze je MySQL a obsahuje jednu tabulku, ve které jsou vlastnosti a informace o vozidlech. Internetový server se nachází na doméně www.vitazman.cz. Data pro plnění databáze, jsou získávána extrakcí HTML dokumentů. Extrakci dat zajišťuje skript `getdata.py` a Framework Scrapy. Extrahovaná data se poté nahrají do databáze. Komunikace mobilní aplikace s databází je implementována použitím PHP skriptů.

V následujících odstavcích popisuji vytvořenou databázi, její tabulku a způsob komunikace s mobilní aplikací.

3.1 MySQL a PHP

Pro práci s daty jsem zvolil relační databázi MySQL. MySQL je nejrozšířenější multiplatformní databáze. Je dostupná na většině serverů a má obsáhlou dokumentaci. Komunikace s databází probíhá pomocí dialektu jazyka SQL s některými rozšířeními. MySQL je k dispozici pod dvojí licencí, bezplatnou licenci GPL a komerční placenou licencí. [2].

PHP je skriptovací programovací jazyk. Je nezávislý na platformě a používá se pro programování webových aplikací a, dynamických internetových stránek. Pro účely této práce je PHP použit pro komunikaci s MySQL databází, protože je velmi rozšířen na serverech a tím pádem má velkou uživatelskou podporu. [3].

3.2 Databáze

Tato část se zaměřuje na popis databáze, a komunikaci s mobilní aplikací. V tabulce `cars` se nachází veškeré informace o vozidlech a jsou popsána jednotlivá pole, jejich datový typ a klíče podle kterých je vyhledána hodnota a uložena do příslušného pole.(Tabulka 2.1).

Tabulka 2.1: Tabulka `cars`

Pole	Datový typ	Klíče, ve kterých se nacházejí hodnoty
cars_id	Int	Jedná se o primární klíč databáze. Slouží k jednoznačné identifikaci vozidla. Hodnota je automaticky generována databází atributem `AUTO_INCREMENT`.
id_sauto	Int	Identifikační číslo vozidla na serveru Sauto.cz. Hodnota je získaná z objektu JSON, pod jménem `advert_id`.
znacka	Varchar	Značka vozidla. Je uložena pod jménem `manufacturer_name` v JSON objektu.
model	Varchar	Model vozidla. Je v JSON objektu uložen jako `model_name`.
typ	Varchar	Typ vozidla - většinou označuje objem vozidla např.: 1.6. V JSON objektu je uložen pod jménem `advert_name`.
stav	Varchar	Určuje, jestli je vozidlo nové nebo ojeté. V JSON objektu je uložen pod proměnnou `advert_condition_cb`.
tachometr	Varchar	Udává počet najetých kilometrů. V JSON objektu je uložen pod názvem `advert_tachometr`.
barva	Varchar	Barva vozidla. V JSON objektu je uložena pod jménem `advert_color_cb`.
Palivo	Varchar	Určuje typ paliva, které, vozidlo spaluje. V JSON objektu je uložen pod jménem `advert_fuel_cb`.
cena	int	Cena vozidla, v JSON objektu je uložen pod proměnnou `advert_price_sale`.
odkaz	Varchar	Odkaz na stránku s kartou vozidla. Je složen z několika dílčích částí, první část je http://www.sauto.cz/, další je uložena v JSON objektu jako `advert_url` a poslední část je v JSON objektu uložena jako `advert_id`. Datový typ varchar.
vin	Varchar	Jednoznačný identifikátor motorových vozidel. VIN se v JSON objektu nenachází a proto je nutné použít spider vin_sp.
kw	int	Udává výkon vozidla a v JSON objektu se nenachází. Pro

		získání této hodnoty jsem použil spider vykon_sp.
rokv	year	Rok výroby vozidla, v JSON objektu je uložena pod jménem `advert_made_date`.
inzerat_vlozen	Varchar	Datum, kdy byl do databáze Sauta.cz vložen inzerát, v JSON objektu je uložen pod jménem `advert_since`.
info	text	Informace o vozidle např.: další výbava, závady a poškození. V JSON objektu se nenachází, pro získání této hodnoty jsem použil spider info_sp.
expired	boolean	udává, zdali se inzerát na serveru Sauto.cz nachází nebo byl odstraněn. V JSON objektu se nenachází, a pro získání této hodnoty jsem použil spider expired_sp.

3.3 Komunikace s mobilní aplikací

Základem fungování celého systému je automatický robot, který prochází internetové servery a extrahuje informace o vozidlech. Tyto informace jsou ukládány do databáze implementované na lokálním serveru. Na tomto serveru se nachází skripty pro komunikaci s mobilní aplikací. Díky těmto skriptům, lze prohlížet data extrahovaná robotem. Skripty jsou napsány v jazyce PHP. Každý skript se připojí do databáze, provede dotaz a výstup zformátuje do formátu JSON.

3.3.1 Seznam vozidel

Tento skript slouží k získání dat z databáze na základě zadaných parametrů uživatele a jeho název je `Jsonscript.php`. Parametry jsou značka, cena, VIN kód, id vozidla, model vozidla a rok výroby vozidla. Mohou být zadány všechny parametry, nebo jen jeden. Aplikace odešle na server požadavek, který se skládá z adresy serveru, názvu skriptu a jeho parametrů. Pro zobrazení vozidel Ford vyrobených v roce 2010 bude požadavek vypadat takto: localhost/jsonscript.php?znacka=ford&rokv=2010.

Abych zjistil, které parametry jsou nastaveny a které nikoliv, používám funkci `isset`. Pokud je parametr nastaven, uloží se jeho hodnota do proměnné, v opačném případě je proměnná NULL. Dotaz pro výběr vozidel vypadá takto:

```
$sql = "SELECT * FROM cars WHERE  znacka LIKE '%$znacka%' AND cena
LIKE '%$cena%' AND vin LIKE '%$vin%' AND rokv LIKE '%$rokv%' AND
cars_id LIKE '%$id%' AND model LIKE '%$model%' order by
`inzerat_vlozen` desc";
```

K vyhledávání podle vzoru ve sloupci používám operátor `LIKE`. Ten porovnává hodnotu ve sloupci v databázi se zadaným řetězcem, který má na začátku a na konci znak „`%`“. Ten se používá k definování zástupných znaků. To znamená, že jsou vybrány řádky, které obsahují zadaný řetězec [3].

3.3.2 Historie ceny

Script `History.php` používá pouze parametr vin. Podle tohoto parametru budou vybrány řádky vozidel, která mají stejný VIN kód a z těchto řádků budou následně vybrány pouze hodnoty datum, kdy byl inzerát vložen a cena, která byla zadána. Takto se zobrazí historie cen vozidel seřazených vzestupně podle data vloženého inzerátu. SQL dotaz vypadá takto: `$sql = "SELECT` `` `inzerat_vlozen`,`cena` FROM `cars` WHERE `vin` = '$vin' order by `inzerat_vlozen` asc";``

3.3.3 Podobná vozidla

Tento skript se jmenuje `podobne.php` a slouží k výběru podobných vozidel na základě vstupních informací. Tyto informace se skládají z parametru značka, model, rok výroby vozidla a aktuální rok. Dotaz SQL: `$sql = "SELECT * FROM cars WHERE znacka LIKE '%$znacka%' AND model LIKE '%$model%' AND rokv BETWEEN '$rokv' AND '$rok'";`

Takto formulovaný dotaz umožňuje napadení databáze vsunutím kódu přes neošetřený vstup a vykonání pozměněného SQL dotazu. Jelikož se zatím neočekává ostré nasazení aplikace, není obrana proti SQL injection implementována.

4 Realizace automatického robota

V této kapitole je popsána funkce automatického robota, který je použit pro získávání informací z webových stránek, nahrávání získaných informací do databáze a četnost spouštění robota.

4.1 Python

Python je dynamický objektově orientovaný skriptovací programovací jazyk. Je vyvíjen jako open-source. Python nabízí instalační balíky pro platformy Linux, Windows, Mac a BSD. Byl navržen tak, aby umožňoval tvorbu rozsáhlých a plnohodnotných aplikací, včetně grafického uživatelského rozhraní. Python je hybridní jazyk, používá objektové, procedurální a funkcionální programování. Jednoduchost a čistota syntaxe dělá Python vhodným programovacím jazykem pro začátečníky. Nabízí významnou podporu k integraci s ostatními jazyky a nástroji a přichází s mnoha standardními knihovnami [4].

4.2 Scrapy

Scrapy je Framework, který slouží k procházení internetových stránek a, k následnému extrahování strukturovaných dat. Může být použit pro mnoho účelů, od shromažďování dat za účelem monitorování až po automatizované testování. Je vytvořen v programovacím jazyku Python a může být implementován na platformách Linux, Windows, Mac a BSD. Je licencován jako open-source [5].

4.3 Instalace Scrapy

Instalace na systém Linux je jednoduchá, stačí zadat do terminálu:

```
pip install Scrapy
```

nebo :

```
easy_install Scrapy
```

Instalace na systém Windows je na rozdíl od systému Linux složitější. V první kroku jsem nainstalování Python 2.7. Poté jsem v systémové proměnné PATH nastavil cestu ke složce, ve které je nainstalován Python 2.7. V dalším kroku jsem instaloval Win32 OpenSSL a opět nastavil cestu do systémové proměnné PATH. Nakonec jsem nainstaloval balíky pywin32, Twisted, lxml, pyOpenSSL a zope.interface [6].

4.3.1 Pywin32

Pywin32 je verze pro Python 2.5, navržena pro spouštění na 32bitových počítačích se systémem Windows. Obsahuje balík modulů, které jsou určeny pro práci pod Windows. Obsahuje

modul pro práci s WinAPI, pro práci s registry, technologií COM, WIN network API a další. Tento balík také obsahuje vývojové prostředí PythonWin [7].

4.3.2 Twisted

Twisted je asynchronní síťová Framework napsaný v Pythonu. Klade důraz na události založené na programování síťových aplikací a multiprotokolovou integraci. Obsahuje platformu pro vývoj internetových aplikací pro Python, podporující mnoho protokolů jako například TCP, UDP, SSL/TLS, IP Multicast. Twisted je založen na událostmi řízeném programování. Obsahuje webový server, chat servery, poštovní servery a další. Twisted je licencován jako MIT License [8].

4.3.3 Lxml

Lxml je knihovna pro zpracování XML a HTML v jazyce Python. Je jedinečná v tom, že kombinuje rychlost a kompletní vlastnosti knihoven libxml2 a libxslt s jednoduchostí Python API, která je většinou kompatibilní ale nadřazená známé ElementTree API. Má stále silný výkon při analýze velkých souborů a několik programovatelných rozhraní. Dokáže analyzovat neplatné XML dokumenty, pokročilé XPath a CSS selektory. Lxml má rozhraní podobné ElementTree a pro základní využití může být lxml použit jako vestavěná náhrada za ElementTree a zvýšit tak rychlost analyzování. Knihovna lxml je licencována jako BSD a knihovny libxml a libxslt jako MIT licence [9] [16].

4.3.4 PyOpenSSL

Modul PyOpenSSL slouží pro komunikaci s OpenSSL. Je vytvořen jako obal pro knihovnu OpenSSL, protože většina metod objektů nedělá nic jiného, než volá odpovídající funkce v knihovně OpenSSL. OpenSSL je implementací protokolů SSL a TSL. Vytvoří bezpečné spojení mezi klientem a serverem. Bezpečné znamená, že spojení je šifrované a chráněné před odposlechem. To také umožňuje ověřit identitu serveru. Důvod pro vznik tohoto modulu byl ten, že podpora SSL v jazyce Python byla silně omezená [10].

4.3.5 Zope.interface

Zope je objektově orientovaný aplikační server, kteýá je naprogramován v jazyce Python. Základními prvky Zope jsou transakční objektová databáze, obsahující vlastní webový obsah, dynamické šablony pro generování HTML, indexy pro vyhledávání, skripty a další. Základní funkcionalitu Zope rozšiřuje mnoho plug-in modulů, které přinášejí nové druhy dokumentů, konektorů do externích úložišť dat a webové aplikace sloužící k různým účelům. Zope.interface je balíček který obsahuje implementaci objektové rozhraní pro Python. Rozhraní je mechanismus, který slouží pro označování objektů jako vyhovující dané API [11].

4.4 Způsob získávání dat

Získávání dat ze stránek inzertních serverů je rozděleno na několik částí. V první části získávám JSON data z AJAX výstupu a nahrávám do databáze. V další části spouštím jednotlivé Scrapy skripty pro doplnění chybějících informací.

AJAX je technologie, která se používá v interaktivních internetových aplikacích. Tato technologie dovoluje měnit obsah svých stránek, umožňuje překreslení celé stránky bez nutnosti použití tlačítka refresh, také podporuje tzv. nekonečné scrollování. To znamená, že když se uživatel dostane na konec stránky, načte se další blok webové stránky. Změna obsahu je realizována pomocí asynchronního zpracování webových stránek. Technologie AJAX má potenciál zredukovat množství přenášených dat a tím snížit zátěž na webové servery a síť.

JSON je jednoduchý textový formát pro výměnu dat. Jde o univerzální datové struktury a téměř všechny moderní programovací jazyky JSON podporují. Je založen na podmnožině programovacího jazyka JavaScript. Je lehce čitelný i zapisovatelný člověkem a snadno generovatelný i analyzovavetlný strojově. Struktura JSON objektu je realizována jako kolekce páru název/hodnota.[15]

4.4.1 Získání JSON objektu

Vozidla na inzertním serveru www.sauto.cz jsou získávána dotazovací metodou GET, ve které jsou definovány různé parametry vozidel, která chceme zobrazit. V mém případě to jsou vozidla nová, ojetá a předváděcí. Tato metoda vrací JSON objekt. Ten obsahuje informace o 15 vozidlech, které jsou zobrazeny na jedné stránce. URL, které odesílá požadované parametry a vrací JSON objekt jsem zjistil pomocí doplňku Firebug pro prohlížeč Firefox.

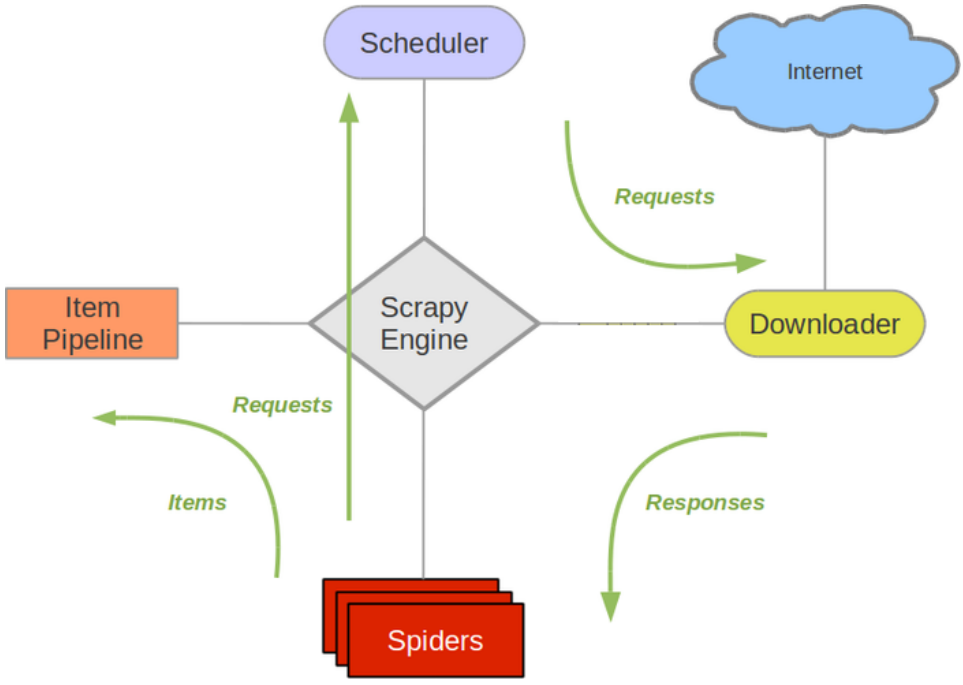
V mém skriptu getdata.py je URL vracející JSON objekt rozdělena na dvě části protože obsahuje parametr `page`, který inkrementuji. Tímto krokem procházím všechny stránky a získávám jejich JSON objekt. V tomto cyklu uložím do proměnné `json_string` JSON objekt stránky. Proces ukládání je realizován díky modulu urllib2 a jeho funkci `urlopen(url)`, která otevře ULR `url` a další funkcí `read()` načte obsah URL. V následujícím kroku se provede funkce `json.loads()`. Tato funkce převede JSON objekt na speciální datový typ slovník neboli asociativní pole. Slovník je datový typ podobný poli, ale každá položka se skládá z dvojice klíč-hodnota. Díky tomu je vyhledávání v poli snadnější.

Následuje vnořený cyklus, který se opakuje počtem vozidel obsažených v JSON objektu. Prochází celým slovníkem a díky klíčům snadno vyhledá požadované hodnoty a uloží je do proměnných. Na konci cyklu funkce insert vloží proměnné do databáze [4].

4.5 Scrapy a extrahování informací

Jelikož skript getdata.py nedokáže extrahovat všechna data, používám Framework Scrapy. Informace, které je potřeba vyextrahovat jsou VIN kód, výkon motoru, informace jako například další výbava vozidla a již expirované inzeráty. Jednotlivé spidery se spouštějí postupně a každý má za úkol extrahovat odlišné informace. V následujícím odstavci popisuji stručnou architekturu Frameworku Scrapy.

Engine frameworku scrapy je zodpovědný za ovládání toku dat mezi všemi komponenty systému. Scheduler přijímá požadavky z enginu a přidává je do seznamu pro pozdější odeslání, když si je engine vyžádá. Downloader je zodpovědný za načítání internetových stránek odesílá je do enginu, který je odešle spideru. Spider je vlstní třída napsaná uživatelem určená k analyzování odpovědí z downloaderu a extrahování požadovaných položek. Item Pipleline je zodpovědný za zpracování již extrahovaných položek.



Obrázek 4.1: architekturu Frameworku Scrapy

V následujících odstavcích popisuji způsob získání těchto informací.

4.5.1 Spider

Spider je třída, která definuje, které stránky budou procházeny, jakým způsobem budou následovat odkazy a jak přesně má extrahovat data. Spider obsahuje metodu `start_requests` a metodu `parse`.

První požadavky pro provedení jsou získány metodou `start_requests`, která generuje požadavek `Request` pro URL stáhnuté z databáze a metoda `parse` pro zpětné volání požadavků. Funkce zpětného volání analyzuje webovou stránku a vrací objekt Item. Analýza probíhá použitím selektorů. Ty vybírají určitou část HTML dokumentu specifikovanou buď XPath nebo css výrazy.Selectory používají knihovnu lxml nebo BeautifulSoup.

Nakonec jsou extrahovaná data vrácena a uložena do objektu Item. Ten může byt exportován do formátů JSON, CSV, XML a mnoho dalších, nebo lze data odeslat do databáze [6].

4.5.2 Vin_sc

Tento mnou vytvořený spider používám pro extrahování a nahrání chybějící informace o VIN kódu. Metoda `start_requests` se připojí do databáze a vybere z tabulky odkazy řádků, ve kterých chybí hodnota ve sloupci vin. Seznam odkazů tvoří generátor, který vrací odkaz jeden po druhém funkcí `yield`.

V metodě parse jsem implementoval funkci `rpartition('/')`, která rozdělí aktuální `current_url` na segmenty, podle nastaveného oddělovače. V mém případě `'/'`. V dalším kroku tak získám z odkazu identifikační číslo vozidla, které dostane od serveru Sauto.cz. Tuto hodnotu pak uložím do proměnné `idsaut`.

V dalším kroku metoda `parse` extrahuje data pomocí selektoru XPath. Ten podle uzlů v HTML dokumentu vyhledá třídu `span`, která má jméno `vin_detail`, a vybere z ní text. Selector pro extrakci VIN kódu vypadá takto: `"//span[@class='vin_detail']/text()"`.

Získaná hodnota je uložena do pole `item['VIN']` a je nahrána do databáze. Na databázi je položen dotaz `UPDATE`, který změní hodnotu `vin` na řádku, jehož hodnota `id_sauto` se rovná hodnotě uložené v proměnné `idsaut`, která byla získána při rozdělení `current_url` na segmenty[6].

4.5.3 Vykon_sc

Spider vykon_sp extrahuje informace o výkonu motoru vozidla. Funguje na stejném principu jako vin_sc, jen s několika rozdíly.

Ten první je dotaz položený na databázi pro zjištění, kterým řádkům chybí informace o výkonu motoru a získání odkazu na inzerát na server Sauto.cz. Opět se vytvoří seznam odkazů a generátor, který vrací odkaz jeden po druhém pomocí funkce `yield`.

Déle pokračuji metodou parse. V této metodě jsem musel změnit selektor XPath. Server Sauto.cz prezentuje hodnota jako objem, převodovka, výkon a další informace v tabulce. Ovšem ne všechny vozidla mají vyplněny všechny informace a tudíž nemusí tabulka vypadat u všech vozidel stejně. Tím pádem nelze v selektoru nastavit pevný řádek pro extrahování hodnoty. Mé řešení spočívá v tom, že metoda `parse` podle uzlů v HTML dokumentu vyhledá tabulku `detailParams` a v prvním sloupci vyhledá řádek Výkon, pomocí XPath funkce `contains`() a z vedlejší buňky získá hodnotu. Selector pro vyhledání výkonu motoru vypadá takto: `"//table[@id='detailParams']//tr[contains(., 'kon:')]/td/text()"`. Vyhledávací řetězec `'kon:'` je z důvodu problému kódování stránek.

Poslední změnou v kódu je pole `item['vykon']` a úprava dotazu `UPDATE`, kde jsem změnil sloupec který, který je aktualizován [6].

4.5.4 Info_sc

Tento spider extrahuje další informace o vozidle, další výbavu. Je implementován stejně jako předchozí dva spidery. Největší problém nastal při extrakci dat. Na serveru Sauto.cz je každá jedna část výbavy zapsána jako jedna třída `span` se jménem `more`. Toto způsobilo, že při extrakci dat se ze získaných hodnot vytvořila tabulka. A extrakce neproběhla v pořádku. Proto jsem v selektoru použil funkci `concat`(), která spojí všechny získané stringy do jednoho. Nakonec jsem opět změnil pole `item['info']` a dotaz `UPDATE` na databázi [6].

4.5.5 Ex_sp

Spider Ex_sp zjišťuje, jestli je inzerát aktuální nebo už vypršela jeho platnost. Scrapy filtruje neúspěšné odpovědi HTTP, takže se spidery zabývají jen úspěšnými odpověďmi. Ty jsou v rozsahu 200-300. Pro zpracování odezvy mimo tento rozsah jsem musel nastavit, které kódy odezvy bude spider schopen zpracovat. Toto jsem implementoval nastavením atributu `handle_httpstatus_list`. Znamená to, že pokud spider chce extrahovat stránku, jejíž platnost už vypršela, následuje vrácený odkaz: http://www.sauto.cz/expirovany-inzerat?id= + identifikační číslo vozidla.

Spider vybere z databáze všechny odkazy řádků, které mají hodnotu `expired` rovnu 0. Pokud je inzerát prošlý, je přeměřován na informační stránku a z této stránky je extrahován řetězec. Poté je extrahovaný řetězec uložen do pole `item['EXP']`. Pokud se rovná řetězci uloženému v poli `item['SE']`, změní se v poli `expired` hodnota na 1 [6].

4.6 Automatizace robota

Aby byla zaručena kontinuita extrahování dat z internetu, napsal jsem krátký dávkový soubor, který se jmenuje RunScripts.bat. Tento soubor spouští skript getdata.py a dále přechází do nejvyšší úrovně adresářů, ve kterých se nachází spidery. Ty jsou spuštěny jeden po druhém pomocí příkazu `CALL scrapy crawl` a název spideru.

Pro nastavení opakování spouštění dávkového souboru jsem vybral nástroj Plánovač úloh. Tento nástroj, který je součástí operačního systému Windows, je schopen naplánovat spuštění libovolného skriptu, programu nebo dokumentu. V tomto nástroji jsem vytvořil Základní novou úlohu. Zadal název úlohy a její popis. V dalším kroku jsem určil, kdy má být úloha spuštěna. Zvolil jsem tedy týdenní cyklus. Nastavil jsem datum a hodinu prvního spuštění a zvolil pondělí jako den opakování. Dále jsem ze seznamu vybral Spuštění programu a zadal cestu k dávkovému souboru. Tímto způsobem realizuji automatického robota pro extrakci dat.

5 Návrh a implementace klientské aplikace pro OS Android

V této kapitole popisuji požadavky aplikace, všechny její funkce, princip stahování a zobrazování informací.

5.1 Cíl aplikace

Cílem aplikace je vytvoření klienta pro zobrazení vozidel uložených v databázi. Uživatel navštíví autobazar a líbí se mu určité vozidlo. Aby zjistil, jaké jsou ceny v jiných autobazarech a další informace použije mou aplikaci. Do vyhledávacích polí zadá hodnoty vybraného vozidla a zobrazí se seznam vozidel. Pak uživatel vybere jedno a zobrazí se všechny dostupné informace.

Z informaci uvedených výše je zřejmé, že aplikace by měla být jednoduchá na ovládání a neměla by zbytečně zatěžovat datové přenosy. Také by měla zobrazovat aktuální informace a nástroje pro zobrazení historie ceny a zobrazení podobných vozidel.

5.2 Technické požadavky a funkce

Aplikace se jmenuje Pre-Owned Cars, což ve volném překladu do češtiny znamená Ojetá auta. Aby mohla aplikace fungovat, musí být na mobilním zařízení nainstalován Android minimálně verze 2.3.3 a také musí být navázáno připojení k internetu.

Pokud je uživatel připojen k internetu může použít aplikaci. Na první obrazovce má na výběr z vyhledávání vozidel, informace o aplikaci a návod na její použití.

Po kliknutí na tlačítko Vyhledávání vozidel se uživateli zobrazí formulář pro zadání informací o vybraném vozidle. Uživatel může vložit informace do všech polí nebo do jednoho pole. Pokud do polí nic nezadá, zobrazí se upozornění, že jsou všechna pole prázdná a je požádán, aby zadal alespoň jednu hodnotu.

Po zadání uživatelem a kliknutí na tlačítko Odeslat, se zobrazí informační zpráva o stahování dat (Obrázek 4.1). Po stáhnutí informací se zobrazí obrazovka s pohyblivým seznamem vozidel. Kromě značky a typu vozidla také zobrazuje cenu, počet najetých kilometrů, rok výroby a výkon motoru. Po kliknutí na vybrané vozidlo se zobrazí karta vozidla.

Na této obrazovce lze vidět všechna dostupná data, která se nacházejí v databázi. Dále zde uživatel nalezne tlačítka pro zobrazení historie ceny vozidla a podobná vozidla.

5.3 Android SDK

Android SDK je vývojový balíček, který je nutné si stáhnout a nainstalovat pro tvorbu aplikace pro Android. Android SDK obsahuje potřebné nástroje pro psaní, testování, ladění aplikací, a API knihovny pro tvorbu aplikací. Balíček také obsahuje SDK manažer, který umožňuje stahovat další knihovny.

K ovládání prvků, které jsou na zařízeních s Androidem, se používají knihovny a nástroje. Jedná se o přístup k sítím pro telefonování nebo práci s daty, přístup k Wi-Fi, GPS a Bluetooth knihovnám. Správa procesů a paměti, podpora multimédií, práce s 2D a 3D grafikou a mnoho dalších. K testování aplikací v počítači slouží Android Emulátor, který vytváří AVD a pomáhá při vytváření aplikace a jejím testování bez použití fyzického zařízení. AVD je virtuální zařízení se systémem Android, ve kterém se spouští vytvořená aplikace. AVD se vytvoří a spravuje v AVD manažeru.

Android SDK a vývojové prostředí Eclipse lze stáhnout v jednom balíku ADT. Je vhodný pro začínající vývojáře, protože obsahuje všechny potřebné moduly pro tvorbu aplikace a jeho instalace je velmi jednoduchá. Balík ADT je freeware a lze ho zdarma stáhnout na developer.android.com [12].

5.4 Uživatelské rozhraní

Základním prvkem každé aplikace je její uživatelské rozhraní. Je to důležitá část aplikace a mělo by být jednoduché, přehledné a srozumitelné. Důležitý je také první dojem, který uživatelské prostředí vytváří. Pokud bude uživatel zmaten a ovládání pro něj bude nepříjemné a nesrozumitelné, aplikaci nebude používat a vymaže ji.

Základní stavební komponentu pro uživatelské rozhraní představuje třída `View`. Slouží pro vykreslení a zpracovávání událostí, a také jako základní třída pro podtřídy zvané `Widgets`. `Widgets` jsou interaktivní prvky uživatelského rozhraní, například tlačítka, seznamy nebo textová pole.

Třída `Activity` představuje prezenční vrstvu vizuální komponenty, se kterou bude pracovat uživatel. Každá třída `Activity` reprezentuje obrazovku aplikace. Do podtřídy `Activity` se vkládají stavební bloky, které budou zobrazeny na obrazovce. Každá třída `Activity` má povinnou metodu `oncreate`. Tato metoda při vytvoření `Activity` většinou inicializuje používané prvky. Třídy dědící z `Activity` musí mít layout.

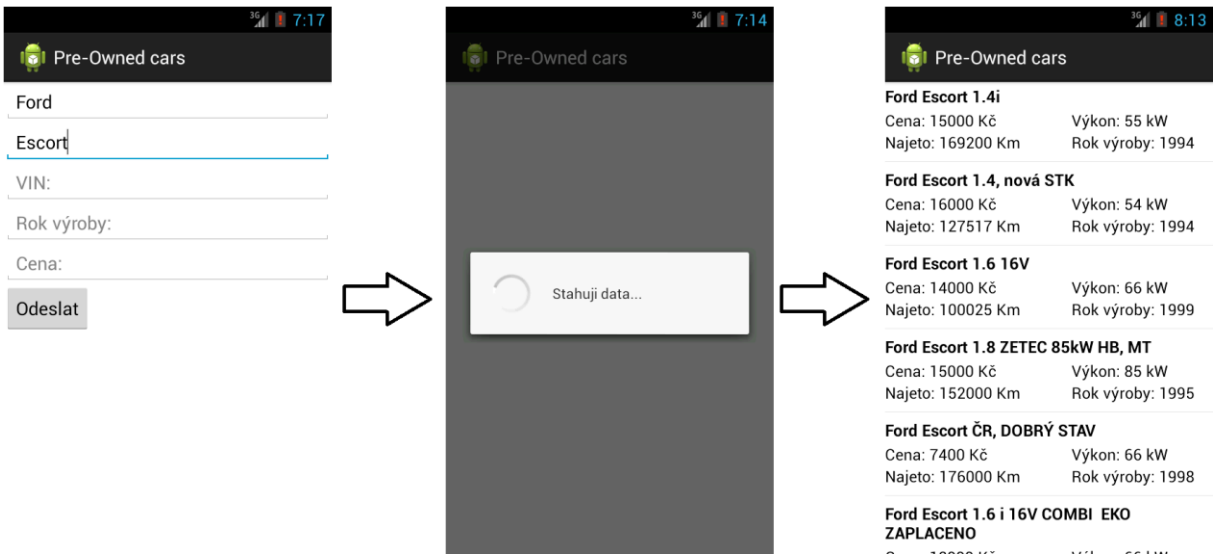
Třída `layout` nabízí rozvržení architektury aplikace, například lineární nebo relativní. Základní třídou pro podtřídy `layouts` je třída `ViewGroup`. `Layout` je XML soubor, který ukládá rozvržení architektury aplikace. `Layout` se nachází ve struktuře Android projektu v adresáři `res/`.

Aby se layout zobrazil na obrazovce zařízení, je nutné přiřadit aktivitě určený layout. Toto se provede v metodě onCreate(), zavoláním metody setContentView(), do které se nastaví požadovaný layout. Pro komunikaci a přecházení mezi jednotlivými `Activity`, se používá třída `intent`. `Intent` umožňuje předávání a odebírání dat. Přidání dat probíhá pomocí metody `putExtra()` a odebrání dat metodu `getExtra()` [13].

5.4.1 Použité prvky uživatelského prostředí

V aplikaci jsem implementoval důležité ovládací prvky, které jsou součástí knihoven Android SDK. Všechny prvky se nacházejí v XML souborech, nazývajících se `layout`. Každý `layout` přísluší určité `Activity`. Aby bylo možné rozeznat jednotlivé prvky a následně je použít v `Activity`, mají vlastní ID. Téměř každý aspekt ovládacího nebo zobrazovacího prvku je nastavitelný pomocí XML kódu, například mezery od kraje obrazovky nebo ostatních prvků či nastavení rozměrů prvků.

Na první obrazovce používám `LinearLayout`, `ImageView` a `Button`. `ImageView` je pole, které slouží k zobrazení obrázku. `Button` je tlačítko. U tlačítek implementuji třídu `OnClickListener`, která zjišťuje, zda uživatel na tlačítko kliknul. Třída `OnClickListener` implementuje metodu `onClick()`. Ta definuje, co se má stát po kliknutí, takže pro přechod do další `Activity` jsem přidal `intent()`.



Obrázek 4.2 *Obrázek 4.1* *Obrázek 4.3*

Průchod aplikačními obrazovkami

`Layout` další obrazovky se jmenuje `activity_search.xml` a slouží jako vstup od uživatele. Proto `layout` obsahuje pole pro editaci textu `EditText`(Obrázek 4.2). Ten umožňuje nastavit velikost zadávaného písma, skrývání znaků pro hesla, zobrazení nápovědy a mnoho dalších možností. Dále `layout` obsahuje tlačítko pro odeslání zadaných údajů.

Obrazovka se jménem `activity_list.xml` implementuje `LinearLayout` a obsahuje `ListView` jménem list. `ListView` reprezentuje vertikální seznam položek, se kterými může uživatel pracovat (Obrázek 4.3). Zobrazení jednotlivých položek je implementováno v `list_item.xml`. Ten obsahuje `TextView` pro zobrazení nadpisu vozidla a `TableLayout`. V `TableLayout` jsou dva řádky a dva sloupce. V prvním řádku je v textovém poli zobrazena cena a výkon, ve druhém řádku je v textovém poli zobrazen počet najetých kilometrů a rok výroby vozidla.

Layout zobrazující kartu vozidla se jmenuje `activity_singlecar.xml` (Obrázek 4.4). Implementuje `LinerLayout a TextView`. Dále obsahuje `ScrollView`, ve kterém je implementován `TableLayout`. `ScrollView` jsem použil z důvodu velkého množství dat. Některá vozidla mají velké množství doplňujících informací a ty se nezobrazí korektně. Proto jsem použil vertikální scrollbar. `TableLayout` obsahuje 9 řádků, ve kterých je implementován `TextView` pro zobrazování informací. Pod `ScrollView` jsou vložena tlačítka. První `button` otevře prohlížeč s odkazem na internetové stránky vozidla. Prostřední `button` zobrazí vertikální seznam položek, který obsahuje datum a cenu vozidla. Poslední `button` opět zobrazí seznam položek s informacemi o vozidlech, která jsou podobná právě prohlíženému vozidlu. Informace jsem čerpal z [13][14].

Obrázek 4.4: Karta vozidla

5.5 Implementace aplikace

V této kapitole se budu zabývat implementací aplikace a popíši problémy, se kterými jsem se setkal.

V aplikaci používám `AsyncTask` pro stáhnutí dat ze serveru a jejich zpracování. `AsyncTask` usnadňuje komunikaci pracovním vláknem a ostatními vlákny a umožňuje vykonávat operace v pozadí. `AsyncTask` by měl být použit pro vykonávání krátkých operací. `AsyncTask` formalizuje životní cyklus vlákna. Ten se skládá ze tří po sobě jdoucích metod. `onPreExecute(),doInBackground(), onPostExecute()`. Při startu AsyncTask se nastaví úloha metodou `onPreExecute(), metoda doInBackground()` se provede hned po ukončení metody `onPreExecute()` a provádí operace v pozadí. Poslední metoda je `onPreExecute()` a ta nastavuje jako svůj parametr výsledky provedených operací.

K zobrazení dat, jsem použil `ListView`, ve kterém jsem implementoval `ArrayList` s prvky `HashMap`. Dalším krokem je implementace `HashMap`, do kterého jsou vložena data. Tento `HashMap` naplněn daty je implementován do `ArrayList`. Pole listů `ArrayList` plní `SimpleAdapter` a ten je nastaven pro zobrazení v `ListView`. `HashMap` je implementace datové struktury `Map` a všechny prvky jsou povoleny jako klíče nebo hodnoty.

`SimpleAdapter` slouží k mapování statických dat. Ta mohou být definována v XML souboru nebo mohou být zadána v doprovodném seznamu, například `ArrayList` s prvky `Map`. Všechny položky v `ArrayList` odpovídají jednomu řádku v seznamu. Ve struktuře `Map` jsou uložena data pro každý řádek. SimpleAdapter umožňuje implementaci zobrazení každého řádku a mapování klíčů ve struktuře `Map`.

5.5.1 Navázání spojení se serverem a stáhnutí dat

O stahování a posílání dat se stará třída `HttpClient`. Pro stahování dat je určena třída `HttpGet`, a o posílání se stará třída `HttpPost`. Třída `HttpGet` obsahuje URL webové stránky. Na tuto URL je zaslána metoda GET protokolu Http. Data, která jsou stránkou vrácena, jsou uložena do `HttpResponse` metodou `execute()`. Metoda `execute()` má parametr `HttpGet` a je implementována ve třídě `HttpClient`. Aby bylo možné data použít, musí být převedeny na datový objekt. To se provede použitím metody `getEntitity()` implementované ve třídě `HttpResponse`. Získaná entita je dále převedena metodou `toString()` do datového typu `String a` uložena do statické proměnné response, která je vrácena při zavolání `HttpClienta`. Takto získáváme data ze zadané URL v požadovaném datovém typu. Informace jsem čerpal z [12][14].

5.5.2 Zobrazení dat

Aby se mohla data zobrazit, musí být nejprve zadány vstupní informace. Tyto informace zadá uživatel ve vyhledávací obrazovce (Obrázek 4.2). Tato obrazovka je implementována ve třídě `SearchActivity`. Základní řetězec obsahuje URL serveru a cestu k PHP skriptu, který vrací seznam vozidel ve formátu JSON. Metoda `onClick()` předává pomocí třídy `Intent` URL s informacemi o tom, jaká vozidla a s jakými parametry mají být zobrazena. Toho jsem docílil testováním, zda jsou textová pole prázdná nebo ne. Pokud je pole vyplněno, připojí se k základnímu řetězci parametr a jeho hodnota. V opačném případě nepřipojí nic. Pokud není vyplněno alespoň jedno textové pole, zobrazí se upozornění. Jestliže je vyplněno alespoň jedno textové pole, je třídou `Intent` a její metodou `putExtra()` přenesena URL do další `Activity`.

V `ListViActivity` jsem deklaroval JSON uzly a implementoval zobrazení dat vytvořením instance `ListView`. Při vytvoření aktivity se sestrojí nová instance `ArrayList`, obsahující prvky `HashMap`. Dále se při vytvoření aktivity definuje rozhraní pro zpětné volání v případě, že uživatel klikne na položku v `AdapterView`. Poté se zavolá asynchronní úloha `GetCars`.

Asynchronní třída `GetCars` nejdříve získá data ze třídy `Intent` pomocí metody `getStringExtra()` a uloží je do proměnné `url`. V metodě `onPreExecute()` zobrazí zprávu o probíhajícím stahování dat.

V další metodě `doInBackground()` je zavolán `HttpClient` s hodnotou uloženou v `url`, která je získaná metodou `getStringExtra()`. Výstup z klienta je uložen do `jsonStr`. Pokud je `jsonStr` nenulový, převede se a uloží do `JSONObject cars`. Metoda `getJSONArray` vrací hodnotu klíče `car`, pokud existuje a je `JSONArray`. Následuje smyčka, která prochází celým JSON polem a ukládá do proměnných hodnoty nalezené podle zadaných klíčů. Dále se vytvoří dočasný `HashMap` pro jedno vozidlo. Do `HashMap` se vloží metodou `put()`, klíč, kterým se vyhledávaly hodnoty v poli JSON a jeho nalezená hodnota. A celý `HashMap` je vložen do `ArrayList`.

V metodě `onPostExecute()` se doplňují analyzovaná JSON data do `ListView`. Vytvoří se `ListAdapter`, který je mostem mezi `ListView` a daty, která budou zobrazována podtřídou `SimpleAdapter`. `SimpleAdapter` mapuje data z `ArrayList` a implementuje zobrazení v `ListView`. [12][14]

Po kliknutí uživatele na vybrané vozidlo zjistí metoda `onItemClick()`, na kterou položku v `AdapterView` bylo kliknuto, a metoda `getText()` vrátí text, který je zobrazován. Tyto hodnoty jsou převedeny na datový typ string metodou `toString()` a uložený do proměnných. Tyto proměnné jsou vloženy a předány do další `Activity` pomocí `Intent`.

5.5.3 Zobrazení karty vozidla

Při vytvoření `Activity` je zavolána asynchorní třída `GetCar`. Ta v úvodu získá URL z `Intent` metodou `getStringExtra()`. V metodě `onPreExecute()` se opět objeví zpráva o průběhu stahování dat.

V metodě `doInBackground()` stáhne `HttpClient` opět JSON String a ten je převeden do `JSONObject`. V `JSONObject` jsou vyhledávány zadané klíče a jejich hodnoty jsou uloženy do proměnných. Protože stávající vlákno nemůže přistupovat k prvkům uživatelského prostředí, použil jsem funkci `runOnUiThread()`. Tato funkce spustí specifickou akci ve vlákně uživatelského prostředí. To znamená, že uvnitř tohoto vlákna mohu nastavovat textová pole. Do textových polí `TextView` jsem metodou `setText()` nastavil hodnoty vyhledané v `JSONObject`.

Z karty vozidla lze tlačítky přejít do dalších obrazovek. Tlačítko pro zobrazení inzertní stránky vozidla používá konstruktor `Intent`, který umožňuje předávání implicitní akce. Metoda `setData()` použita k poukázání na umístění internetových stránek a metoda `parse ()převede String` na `URI`.

Další tlačítka mají podobnou funkčnost. Předávají data skrze `Intent` a spouští nové `Activity`. Liší se pouze v odkazech, které předávají. Tlačítkem Zobrazení historie ceny, se předává odkaz na skript `history.php` a hodnota identifikačního čísla VIN. Tlačítko Zobrazení podobných vozidel předává odkaz na skript `podobne.php` a přidává informace o značce vozidla, modelu vozidla, roku výroby a aktuálním roce. Obě `Activity` pak vytvářejí `ListView`. Obrazovka s podobnými vozidly je stejná jako první obrazovka s výběrem vozidel (Obrázek 4.3) a obrazovka s Historií ceny používá stejný layout jako ostatní `ListView`, ale využívá jen dvě textová pole.

5.5.4 Android Manifest

Android Manifest je XML soubor, který obsahuje každý vyvíjený projekt pro Android. Definuje chování aplikace a její strukturu, obsahuje výčet aktivit aplikace a služeb spolu s oprávněními a funkcemi, které aplikace potřebuje pro správnou funkčnost. Obsahuje metadata, komponenty a požadavky aplikace. Dále jsou v tomto souboru definováni poskytovatelé obsahu a jsou zde registrovány `intent` filtry aplikace. Android Manifest se také stará o zabezpečení aplikace a definuje hardwarové požadavky. Soubor také deklarativně definuje oprávnění, která využívá aplikace, stejně jako může udělit speciální oprávnění pro jiné aplikace využívajících služeb dané aplikace.

5.6 Řešené problémy při implementaci

Při testování mobilní aplikace jsem narazil na několik problémů. Ty způsobovaly, že se data ze serveru nezobrazovaly korektně. Proto jsem implementoval skript format.py, který níže uvedené problémy obpravuje.

5.6.1 Formátování vstupů do databáze

V první řadě jsem zjišťoval, v jakém formátu se data nahrávají do databáze a jakým způsobem je správně formátovat, aby nevznikal problém při pokládání SQL dotazů. Z toho důvodu jsem musel formátovat datum ve skriptu `format.py`, aby bylo možné řadit data podle data vložení inzerátu a korektně zobrazovat historii ceny.

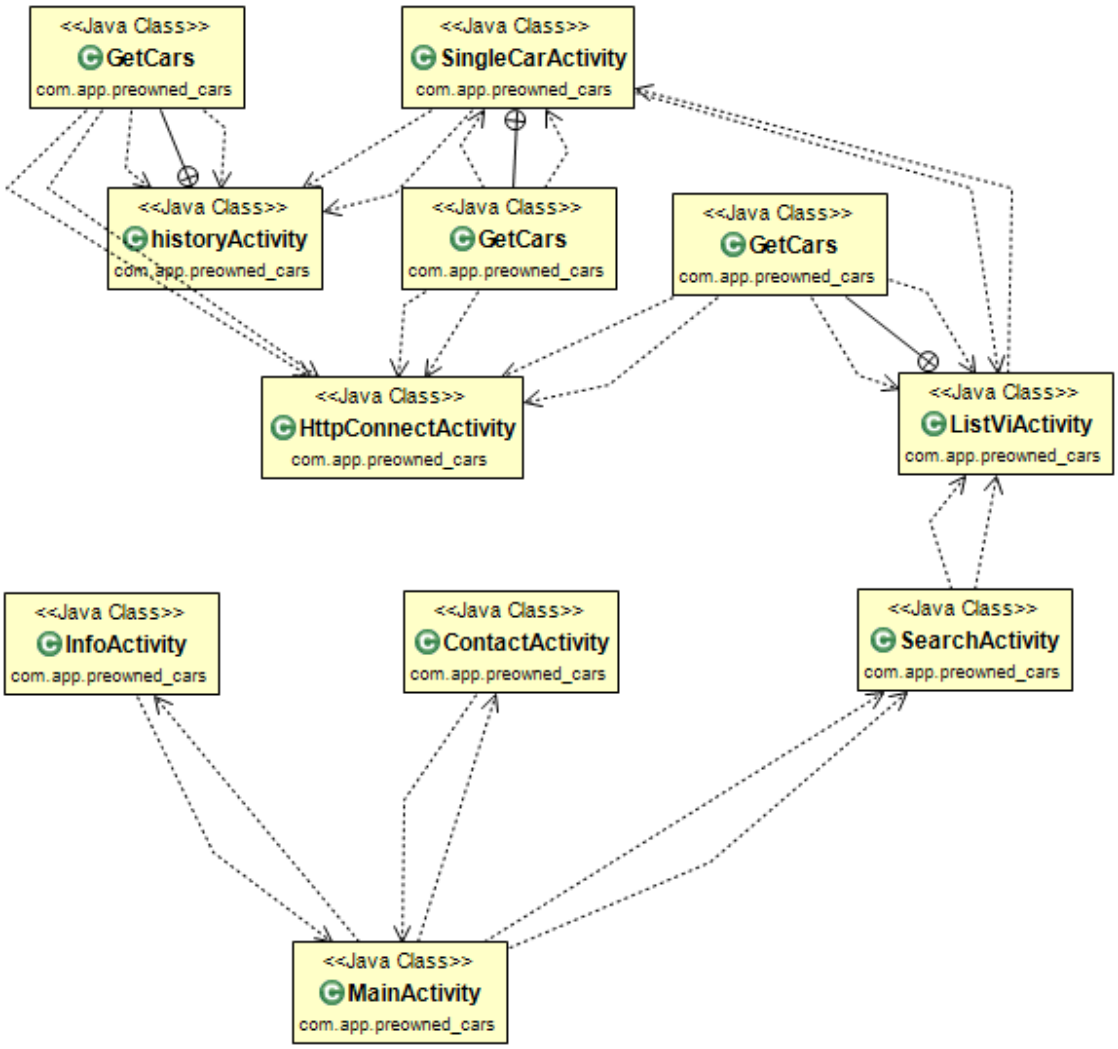
Další problém nastal při selektování vozidel se stejným VIN kódem. Zjistil jsem, že spider `vin_sp` extrahuje VIN kód s mezerou před řetězcem. Proto jsem do skriptu `format.py` implementoval formátovací SQL dotaz, který mezeru v databázi odstraní.

Jedním z velkých problémů bylo obrovské množství duplicitních záznamů v databázi. Toto množství způsobil skript `getdata.py`. Ten během spuštění extrahoval téměř tisíc vozidel a ukládal je do databáze. S každým dalším spuštěním přibývalo do databáze množství dalších vozidel. Vozidla, která byla na inzertním serveru nová ale také i vozidla, která se už v mé databázi nacházela. Proto jsem do skriptu `format.py` implementoval SQL dotaz, který z databáze odstraní vozidla, která mají stejný vin kód a cenu. Následně v databázi zůstanou vozidla, která jsou totožná, ale došlo u nich ke změně ceny, čímž-lze lépe zobrazit historii ceny vozidla. SQL dotaz vypadá následovně:

```
DELETE car1 FROM cars car1, cars car2 WHERE car1.cars_id > car2.cars_id AND car1.vin = car2.vin AND car1.cena = car2.cena
```

5.7 Třídní diagram a použité třídy

Aplikace byla vyvíjena v jazyce Java, který je programově orientovaným jazykem. Základem objektového programování je objekt, který má své atributy a metody. Pro definování vlastností objektů jsou využívány třídy (Obrázek 4.5). Třídy pro zobrazení na obrazovku zařízení jsou `MainActivity`, která zobrazuje úvodní obrazovku (Obrázek), dále `SearchActivity` která zobrazuje vyhledávací obrazovku (Obrázek 4.2). Aktivita `ListViActivity` zobrazuje seznam vozidel (Obrázek 4.3), `SingleCarActivity` zobrazuje kartu vozidla (Obrázek 4.4) a `HistoryActivity` zobrazuje historii ceny vozidla. Třídy `GetCars` jsou `AsyncTack` vlákna a obsluhují komunikaci se serverem. Jejich obrazovka je na Obrázku 4.3.



Obrázek 4.5: Třídní diagram

6 Testy databázové a aplikační části

V této kapitole popisuji postup testování mobilní aplikace a mobilní zařízení, na kterých jsem aplikaci testoval. V dalším odstavci popisuji testování databáze a čas potřebný k její plnění.

6.1 Testování mobilní aplikace

Na začátku vývoje jsem mobilní aplikaci testoval v emulátoru AVD. V první fázi jsem testoval uživatelské rozhraní, funkčnost tlačítek a získávání dat z pole `EditText`. Dále jsem testoval skládání URL pro stáhnutí požadovaných dat a jeho předávání dat nástrojem `Intent`. V další fázi jsem testoval přenos dat mezi aplikací a serverem. Jelikož jsem aplikaci testoval v emulátoru, byla rychlost stahování dat velmi pomalá. Všechny výstupy jsem kontroloval v nástroji pro zachytávání a zobrazování výstupů LogCat. Ve finální fázi vývoje jsem aplikaci testoval nejprve na vlastním mobilním zařízení Motorola RAZR. Když jsem se ujistil o funkčnosti, nainstaloval jsem aplikaci na další zařízení, která jsou zobrazena v tabulce (Tabulka 5.1). V tabulce lze vidět testovaná zařízení, jejich verze operačního systému a rozlišení obrazovky. Jedná se o mobilní telefon Motorola a tablety Nexus 7 a Samsung Galaxy Tab 2. Všechny mobilní zařízení mají novější verze Androidu. Bohužel jsem v mém okolí nenašel nikoho, kdo by měl zařízení s nižší verzí Androida.

Aplikace byla testována v několika bodech. První část byla zaměřena na vykreslování uživatelského prostředí na různých obrazovkách. Dále jsem testoval navázání internetového spojení přes wi-fi i mobilní internet. Nakonec jsem aplikaci testoval v reálném provozu a našel několik podobných vozidel a historii jejich ceny. Lepších výsledků by bylo možné dosáhnout s větším počtem položek v databázi.

Tabulka 5.1: Testované mobilní zařízení

Mobilní telefon	Verze OS	Rozlišení obrazovky
Motorola RAZR XT910	4.1.2	540 x 960
Nexus 7	4.4.2	1920 x 1200
Samsung Galaxy Tab2 10.1	4.2.2	1280 x 800

6.2 Testování databáze

Testování databáze jsem věnoval poměrně dlouhou dobu. Důležité bylo, aby se při plnění databáze extrahovaly správné hodnoty a ukládaly se do správných polí. Abych se ujistil, že spidery

budou extrahovat požadovaná data, vytvořil jsem si v Google Drive tabulku a v ní použil funkci `ImportXML()`. Tato funkce analyzuje zadanou internetovou stránku, v mém případě inzeráty na sauto.cz a Xpath dotaz. Pokud je dotaz zadán správně, zobrazí se požadovaná hodnota. Takto jsem testoval všechny dotazy.

Skript `getdata.py` prochází stránky na serveru Sauto.cz a zastaví se na stránce 67, jelikož již neobsahuje další ojetá nebo nová vozidla. Skript běží průměrné 1 minutu a 45 vteřin a za tuto dobu extrahuje 934 položek. Délka extrahování dat pomocí Frameworku Scrapy závisí na počtu chybějících informací v databázi. Pokud budu mít prázdnou databázi a spustím extrakci dat všech informací ze serveru, poběží skript téměř 5 minut. Z toho vyplývá, že čas potřebný k extrakci Frameworkem Scrapy je v tomto případě 3 minuty 15 vteřin. Důležité je také správné formátování vstupních hodnot. Všechny testy jsem prováděl na počítači s konfigurací Intel Core i5, 4 GB RAM, Windows 8.1 Pro 64bi s rychlostí připojení k internetu 15/1,5 Mbit/s.

Databázi jsem testoval na lokálním i internetovém serveru. Na lokálním serveru jsem implementoval plnění databáze robotem a navázání kontaktu s mobilní aplikací. Na internetovém serveru jsem realizoval pouze komunikaci typu klient-server. Plnění databáze jsem neimplementoval, jelikož k databázi přistupuji přímo a poskytovatel hostingu má tento přístup z venčí zpoplatněn.

7 Závěr

Cílem této bakalářské práce bylo vytvořit systém, který získává data z internetových stránek a zobrazuje je v mobilní aplikaci. Tento systém je složen ze tří částí.

V první části této bakalářské práci jsem se zabýval problematikou získávání informací z internetu a jejich následného použití. Zjistil jsem, že téměř každá informace na internetu se dá jednoduchým způsobem získat a uchovat. Tento proces velmi usnadňuje automatizace, která zkrátí extrakční dobu na řády minut. Ze získaných informací lze dojít k požadovaným závěrům, například k analýze dat. Lze konstatovat, že agregáty informací mají svou budoucnost.

V další části jsem se zkoumal využití databáze a získaných dat. Došel jsem k závěru, že pokud mají být data uchována, měla by být kompletní. Chybí-li důležitá informace, která se k danému problému vztahuje a potřebujeme-li s touto informací pracovat, nemá význam ji v databázi uchovávat.

V závěrečné části jsem vytvářel aplikaci pro mobilní zařízení, která mělo s daty uloženými v databázi pracovat. Aplikace je funkční, jednoduchá, přehledná a uživatelsky příjemná. Nevýhodou aplikace není žádná z jejich funkcí, ale databáze, která obsahuje zlomek vozidel nabízených na internetu. Dle mého názoru by tato aplikace mohla uspět u specifického okruhu uživatelů i mimo něj.

Prostor pro rozšíření celého systému je obrovský. V první řadě bych se zaměřil na agregaci informací. Implementoval bych další automatické roboty pro extrakci dat z dalších inzertních serverů a autobazarů. Dále je zde možnost rozšířit extrakční funkce robota o stahování obrázků vozidel a získávání kontaktů přímo na majitele vozidla. Také by bylo nutné změnit architekturu databáze, aby byla schopna rychle pracovat s velkým objemem dat a implementovat zabezpečovací prvky. Dalším prostorem pro zlepšení je samotná aplikace. Zde bych implementoval více možností pro vyhledávání, našeptávač a je zde možnost zobrazovat obrázky vozidel. Dále bych implementoval prvky sociálních sítí, které by nabízely sdílení vybraného vozidla nebo diskuzi pod vozidly a jejich hodnocení. V aplikaci by také mohly být zobrazeny nejnavštěvovanější inzeráty nebo zobrazování náhodných vozidel. Pokud by celý systém obsahoval vyjmenované prvky, usuzuji, že by mohl být velmi úspěšný.

Při vypracování této bakalářské práce jsem výrazně zlepšil své programovací schopnosti a také jsem lépe porozuměl syntaxi programovacích jazyků. Lépe jsem porozuměl internetovým protokolům a technologiím. Dozvěděl jsem se o struktuře internetových stránek a jak v nich nalézt požadované informace a vyzkoušel jsem si práci s databází a jejími dotazy. Největším přínosem pro mne však bylo, že jsem prošel všemi fázemi vývoje takového systému, a zjistil, že problémů které jsem řešil v průběhu vývoje, bylo mnohem více, než jsem na začátku očekával.

Použitá literatura

[1] Google Play. https://play.google.com/store/apps. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: https://play.google.com/store

[2] MySQL :: About MySQL. http://www.mysql.com/about/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://www.mysql.com/about/

[3] PHP: Hypertext Preprocessor. http://www.php.net/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://www.php.net/

[4] Python v2.7.6 documentation. https://docs.python.org/2.7/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: https://docs.python.org/2.7/

[5] Scrapy | An open source web scraping framework for Python. http://scrapy.org/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://scrapy.org/

[6] Scrapy 0.22 documentation. http://doc.scrapy.org/en/latest/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://doc.scrapy.org/en/latest/

[7] PyWin32 – PyCZ. http://www.py.cz/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://www.py.cz/PyWin32

[8] Twisted. https://twistedmatrix.com/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://twistedmatrix.com/documents/current/core/howto/index.html

[9] lxml – Processing XML and HTML with Python. http://lxml.de/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://lxml.de/

[10] Welcome to pyOpenSSL\'s documentation! – PythonHosted.org. http://pythonhosted.org. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: http://pythonhosted.org/pyOpenSSL/

[11] PyPI – Python. https://pypi.python.org/. [online]. 22. 4. 2014 [cit. 2014-04-22]. Dostupné z: https://pypi.python.org/pypi/zope.interface/4.0.5

[12] Android Developers. http://developer.android.com. [online]. 24.4.2014 [cit. 2014-04-24]. Dostupné z: http://developer.android.com/tools/index.html

[13] Vávrů Jiří. Programujeme pro Android [online]. 2013. [cit. 2014-04-24]. ISBN 978-80-247-4863-4. Dostupné z: books.google.cz/books?isbn=8024748630

[14] Komatineti. Pro Android 4. : Springer, 2012. ISBN 978-1-4302-3930-7.

[15] JSON. http://www.json.org/. [online]. 27.4.2014 [cit. 2014-04-27]. Dostupné z: http://www.json.org/

[16] Python 3 Object Oriented Programming [online]. 2010. [cit. 2014-04-28]. ISBN 1849511276, 9781849511278. Dostupné z: books.google.cz/books?isbn=9781849511278

Seznam příloh

Součástí BP je CD.

Adresářová struktura přiloženého CD:

Mobilní aplikace – instalační soubor .apk

Mobilní aplikace-projekt – Eclipse projekt

Scrapy – skripty k extrakci dat

Server – PHP skripty

SQL – SQL skripty pro vytvoření databáze